SB^{3}C2023
Summer Biomechanics, Bioengineering and Biotransport Conference
June 4-8, Vail, CO, USA

USABILITY OF LOW-COST 3D VISUALIZATION SHARING INTERFACES FOR CARDIOVASCULAR BLOOD FLOW DYNAMICS DATA

Zainab Husain (1,*), Noah Egnatis (1,*), Karol Calò (2), Diego Gallo (2), Umberto Morbiducci (2), Peter Coppin (3), David A. Steinman (1)

(1) Biomedical Simulation Lab, University of Toronto, Toronto, Canada
(2) The Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Torino, Italy
(3) Perceptual Artifacts Lab, Ontario College of Art and Design University, Toronto, Canada
*Co-first authors

INTRODUCTION

Easy access to explore-able blood flow data (i.e., data that can be quickly visualized and adjusted to the user's liking) is crucial for the task of bridging technical gaps between communities and involving researchers from different disciplines in cardiovascular research. Especially when viewing complex 3D flows, and potentially multiple complementary flow properties simultaneously (e.g., isosurfaces and streamlines), it is beneficial, and often necessary, for the viewer to be able to alter the visualization independently, such as adjusting the camera angle and altering the plotting thresholds. Designing visualizations to allow for real-time interaction increases the viewer's agency [1] and more closely mirrors the medical data viewing experiences of clinically oriented research collaborators such as radiologists and medical imaging specialists [2].

While robust engineering 3D visualization tools such as Paraview can be used interactively, setup requires multiple steps, including the downloading of large data files and software; and the data interaction requires familiarity with the software's often-dense interface [3]. Researchers more familiar with medical imaging software can find this challenging or time-consuming [1] and, as a result, it is more commonplace to share 3D visualizations with these researchers as static 2D images due to ease of access (e.g., 2D screenshots through email), at the cost of the user's agency. Certain academic journals such as the Journal of Fluid Mechanics have very recently provided solutions to this problem through cloud-hosted Jupyter notebooks [4]. These solutions involve their own trade-offs, such as the fees associated with cloud-based hosting services and the resulting data size limits. The goal of this pilot usability study was to understand the practical trade-offs of free or minimal-cost strategies for sharing 3D flow visualizations, using helical flow in the aorta as an example use-case [5].

METHODS

An aortic helicity visualization was created from 4D phase-contrast MRI data, and was shared using 3 free or low-cost techniques: a Google Colab notebook, a Docker contained notebook, and a Flask webpage [https://github.com/Biomedical-Simulation-Lab/web-viz]. Biomedical engineering researchers with varying computational/programming skill levels were surveyed after accessing data through each technique. Each technique was evaluated on ease of setup and interactivity for the user, and ease of deployment for the visualization developer (i.e., the authors). The data was preprocessed in Paraview, where velocity streamlines and local normalised helicity (LNH) values were computed and saved as VTK unstructured grid files (*.vtu) files. All techniques utilized the PyVista Python library as their visualization basis.

Technique 1: Google Colab: Google Colab is a type of Jupyter notebook hosted on Google's cloud servers. For setting up shareable visualizations on Colab, the developer must store the data in an online storage service (in our case, GitHub) and simply specify the data location within the notebook. This method is quick and free to deploy but is mainly suitable for lightweight datasets due to the processing limitations available on the free version of Colab.

From a user perspective, a visualization created on Colab can be accessed through a web-link and requires no software setup on the user's local computer. The visualization is displayed as a static 2D image that can be rapidly updated by the user after changing the parameter variables (view angles, LNH threshold, streamlines on/off) directly in the PyVista notebook's displayed code.

Technique 2: Docker (Containerized Notebook): Python visualization scripts typically require installation of dependent libraries and require the user to set up their computational environment. As well, running a Jupyter notebook on a local computer can result in hardware related graphics errors for 3D rendered objects. For this reason, it is considered best practice to run local notebooks within a container environment (here, we used Docker) to standardize visualization rendering for different computers and increase stability [6]. To set up the Docker visualization the developer is required to have knowledge of container-based development, such as creating Docker images (containing the notebook and the data) and pushing the images to Docker Hub.

For the user, this technique requires the most setup time and technical skill, requiring the installation of Docker software, and using terminal commands to start the container before the notebook can be viewed. In our survey example, participants adjusted parameters by editing the notebook code as in the Colab technique. However, the Docker technique has the added benefit of 3D object rotation, as the customizability of Docker allowed us to specify the necessary 3D graphics back-end libraries. There was no cost associated with this method.

Technique 3: Flask Web-page: Flask is a Python framework that can be used to build web applications. Utilizing a web-page to display visualizations requires all of the knowledge of the previous two techniques plus knowledge required to develop a web application (familiarity with HTML and JavaScript) and to deploy the application through a chosen cloud server (e.g. Heroku, AWS). For our survey example, the application was hosted on the Heroku cloud platform for a monthly cost of $5 USD which would cover up to 1000 hours of usage/month.

The user is able to access the visualization through a web-link, rotate the object in 3D, and use intuitive widgets (e.g. sliders, check boxes) for changing thresholds or turning visualization elements on/off.

Table 1: Summary table of visualization techniques for developer (top) and user (bottom).

	Colab	Docker	Flask
Ease of development	Easy	Intermediate	Advanced
Cost for sample dataset	Free	Free	Priced
Rendering location	Server	Local	Server
User setup	Web link	Software download	Web link
Interface	Notebook	Notebook	HTML Webpage
Threshold adjusting	Editable code	Editable code	HTML widgets
Object rotation	replotable 2D	Mouse rotatable 3D	Mouse rotatable 3D

Survey Design: Study participants ranged in educational level from Master's and PhD students, to postdocs and professors. Each participant was surveyed on their prior experience using the data exploration tools. Participants described their familiarly in 4 skill areas (Python, notebooks, terminal commands, and Docker) by categorizing themselves as either Novice (never used this tool), Beginner (occasionally used this tool) and Experienced (comfortable using this tool) for each of the skills. Ease of setup and interactivity was rated for each method on a 0 to 5 Likert scale, with 0 indicating inability to access the data, 1 indicating extreme difficulty and 5 indicating extreme ease. Participants were also asked their overall likelihood of using each method (unlikely, maybe, likely).

RESULTS

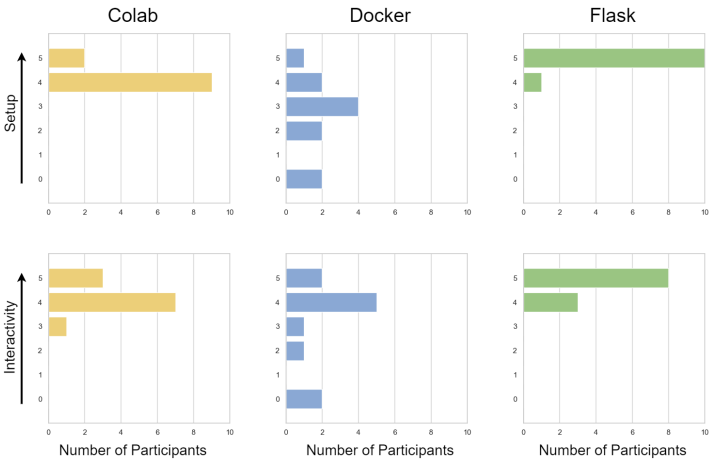

Figure 1: Survey results of setup and interactivity ratings for each data sharing technique.

Participants were generally Experienced at using Python, and Beginners at using notebooks and the terminal commands. Only 1 participant had prior experience using Docker. During the survey, all 11 participants were able to access visualizations shared through Flask and Colab. 2 participants were not able to access visualizations through Docker on their personal laptops due to technical challenges during installation. Flask had the highest average ratings for both ease of setup and interactivity. Docker had the lowest ratings for both categories. However, when asked their likelihood of using each data sharing method, only 4/11 responded likely to use Colab, compared to 7/11 for Docker and 11/11 for Flask.

DISCUSSION

One key question of interest for this study was to compare the preference of mouse rotate-able 3D objects in comparison to a 2D image that can be re-plotted by the user, the latter a low-tech solution which still allows some interactivity. However, the lack of interactive rotatability was reported as the main drawback to Colab by participants during the post survey discussion. While 3D objects do exist in some libraries available in Colab, 3D support is currently unstable and, in particular, is not recommended for dense meshes. We surmise that Google Colab could be a quick and preferred solution for rapid and low-cost sharing between colleagues, but not until there is stable 3D rendering support.

Initial setup of Docker containers was inconvenient for computer beginners due to difficulty installing and configuring on non-Unix based systems. This technique is well suited for situations involving processor-heavy visualizations and limited budget, as all computations are done locally rather than on a server. This places requirements on the user to have a suitable personal computer, which could be a viable option for motivated users for continued use.

Overall, users had a strong preference for quick setup time and rotatable 3D plots, as demonstrated by the clear popularity of the Flask web-page technique. The visual simplicity of the web-page displaying only the data plot and control widgets was also noted as a benefit by participants in comparison to the notebook/code interface, despite that fact the many users were themselves coders. This method also provides a high degree of customizability, and server hours can be scaled up as required. Indeed, during the Flask portion of the survey, multiple users accessing simultaneously caused server failure initially, and participant access had to be staggered. The estimated number of simultaneous users is therefore important to consider when determining computational resources to allocate for a web-page. This method would be the optimal solution for users with minimal technical background, and is a use case where it may be worth investing the extra time, money and effort for the developer.

For this pilot usability study, we desired to have all participants together online, hence it was limited to biomedical engineers due to the scheduling challenges for hospital-based clinical users. Lessons learned from this study will inform future testing with clinical users around the Flask web-page method, and consideration of the cost-benefit of more customizable paid cloud hosted notebook services, such as the Journal of Fluid Mechanics' use of CoCalc [4].

ACKNOWLEDGEMENTS

This work was supported by a grant to DAS from the Natural Sciences and Engineering Research Council (RGPIN-2018-04649). ZH was also supported by a Barbara and Frank Milligan Fellowship.

REFERENCES

[1] Quam DJ et al., J Biomech Eng 2015;137(3).
[2] Temor L et al., Int J Comput Assist Radiol Surg 2022;17(6):11431154.
[3] Schroeder W et al., The vis toolkit. New York: Kitware, 2006.
[4] Meneveau C and Caulfield CP, J Fluid Mech 2022;952.
[5] Morbiducci U et al., Biomech Model Mechanobiol 2011;10:339–355.
[6] Merkel D, Linux J 2014;2014(239).